Probabilistic Model Checking of Ant-Based Positionless Swarming

Paul Gainer, Clare Dixon, and Ullrich Hustadt

Department of Computer Science, University of Liverpool
Liverpool, L69 3BX – United Kingdom
P.Gainer, CLDixon, U.Hustadt@liverpool.ac.uk

Abstract. Robot swarms are collections of simple robots cooperating without centralized control. Control algorithms for swarms are often inspired by decentralised problem-solving systems found in nature. In this paper we conduct a formal analysis of an algorithm inspired by the foraging behaviour of ants, where a swarm of flying vehicles searches for a target at some unknown location. We show how both exhaustive model checking and statistical model checking can be used to check properties that would be impossible to ascertain using simulation alone, resulting in information that would facilitate the logistics of swarm deployment.

1 Introduction

A robot swarm is a multi-robot system comprised of some number of simple, autonomous, homogeneous robots, working together to achieve objectives in some environment without centralised control [16]. Coordination between members of the swarm is achieved through self-organisation and local interactions [2]. The aim of the design of decentralised control algorithms is to produce robot swarms that are *scalable* and *fault tolerant*.

Swarm behaviours are generally analysed through simulation and observations of real implementations. The formal analysis of swarm behaviours can complement the design of swarm algorithms by revealing potential problems that may go unnoticed by empirical analysis [7]. Formal verification is the process by which a property expressed in a suitable formalism (usually some form of temporal logic) is exhaustively checked against every possible run of a system.

Temporal verification has been applied to robot swarms. In [1] deductive verification was applied to prove properties of the foraging behaviour of a swarm of robots; algorithmic verification techniques helped to analyse and refine swarm aggregation in [7]; statistical runtime verification combined with agent-based simulation was used to determine the likelihood of emergent swarm behaviours in [10]; an agent-based temporal-epistemic approach is used in [13] to specify and verify emergence in swarms, and in [12] a probabilistic analysis of population-based swarm models was conducted. Whilst this work has clearly demonstrated that formal verification can be used to exhaustively analyse swarm behaviours, there are still many problems that need to be addressed. Notable issues include

the state space explosion that occurs with naive modelling of swarms, and the need for a general framework applicable to a range of swarm algorithms.

Designing control mechanisms for swarms is a challenging problem. Individual robot behaviours must be formulated at the *microscopic* level and should result in the emergence of complex desired group behaviours at the *macroscopic* level. There are many examples found in nature of decentralised systems that solve complex problems [3]. A common approach in swarm robotics has been to develop control algorithms based on abstractions of these natural systems. In particular, much work has been conducted to develop control algorithms based on the behaviours of social insects, such as foraging for food [5, 14], cooperative nest building [18], and efficient distribution of labour [4].

In [9] a swarm of micro air vehicles (MAVs) attempts to form a communication pathway between multiple ground users in a disaster area. The control algorithm used by the MAVs is inspired by the stigmergic foraging behaviour of army ants which lay and maintain pheromone paths from their nest to sources of food. A model of this behaviour was originally developed in [6], where the results of running Monte Carlo simulations of ants moving through a discrete network of points were analysed. These findings were later discussed in detail in [3].

In this paper we apply probabilistic temporal verification to the scenario presented in [9]. For its verification we generate parameterised formal models for the probabilistic model checker Prism, which we use to either exhaustively or statistically test probabilistic reachability and reward-based properties. While exhaustive model checking checks a property against all possible runs of the system, statistical model checking performs statistical analysis over a subset of the possible runs of the system. We validate our models by comparing the results of checking temporal properties in our models to results obtained from simulations in [9]. We demonstrate how values pertaining to the logistics of deployments of swarms of MAVs, that would be unobtainable through simulation alone, can be calculated a priori by exhaustively checking reward-based properties against every possible execution of our models. This work is an initial step towards the development of a generic probabilistic verification approach that can be applied to other swarm algorithms inspired by the pheromone-based foraging behaviour of social insects.

Section 2 introduces the ant-based swarming scenario described in [9]. In Section 3 we detail the generation of our parameterised input models for the probabilistic model checker Prism, and discuss the abstractions used and assumptions made when designing the discrete formal model. The results of checking probabilistic temporal logic properties in our models are given in Section 4. Concluding remarks and suggestions for further work are given in Section 5.

2 The Ant-Based Swarming Scenario

The scenario to which we apply probabilistic model checking techniques is presented in [9]. Here, a simulated swarm of *positionless* fixed-wing Micro Aerial

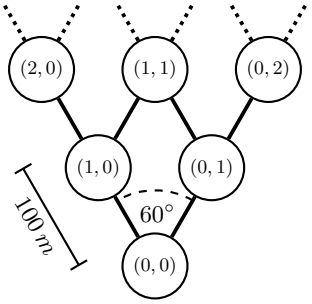

Fig. 1. The Y-junction grid illustrating the ideal positions for MAVs. Each node is 100 m distant from each of its neighbours.

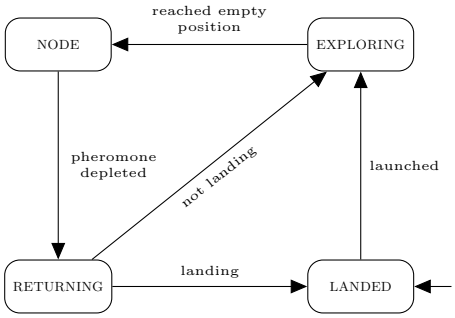

Fig. 2. A finite state machine describing the behaviour of a MAV.

Vehicles (MAVs) is deployed by a human operator in order to establish a robust emergency communication network between a *target user*, situated at some unknown location, and the base station wherefrom the swarm is launched. Each MAV is positionless in that it relies solely upon proprioceptive sensors and local neighbourhood communication to position itself [17]. The establishment and maintenance of this communication network is studied in detail in [9], however in this paper we focus our analysis on the exploration behaviour of the MAVs.

Figure 1 shows a Y-junction grid consisting of possible positions that MAVs will ideally adopt in their search for the target user, and the paths that connect them, while Figure 2 shows the finite state machine describing the behaviour of an individual MAV. A MAV begins in landed state at the *base node*, denoted as $(0, 0)$ on the grid. MAVs are launched at regular intervals and are then in the exploring state. In the exploring state a MAV navigates through the grid, travelling at a velocity of $10\,m/s$. When a MAV reaches a position in the grid where there is no other MAV it will change to node state and remain at that position, acting as a platform upon which other MAVs can "deposit" virtual pheromone. Upon changing to node state a MAV will initialise its pheromone levels to some given amount. A MAV in the node state at some position (i, j) is considered to be an *internal* node if there is a MAV in the node state at either of $(i{+}1, j)$ or $(i, j{+}1)$. Internal nodes supplement their levels of deposited pheromone at each time step by some amount. While in the node state each MAV broadcasts its pheromone level to other MAVs within its communication range of $100\,m$. When a MAV in the exploring state reaches a position in the grid where there is already a MAV in the node state, it continues moving outward and makes a probabilistic choice which branch to take, determined by the levels of pheromone deposited at the next positions on the left and right branches, as transmitted by the MAV in the node state.

Pheromone levels dissipate gradually over time and when they are depleted a MAV in the node state changes to returning state (internal nodes are supplemented with sufficient pheromone to guarantee that they cannot change to

the returning state.) It then navigates back through the grid towards the base node similarly to a MAV in the exploring state but only moving along positions occupied by MAVs in the node state. Once it reaches the base node, if a signal to land is being broadcast by the base node then the MAV will land, otherwise it will change back to exploring state.

In more detail, the choice between the left and right path from some position (i, j) is determined probabilistically according to the amount of deposited pheromone at $(i{+}1, j)$ and $(i, j{+}1)$ for MAVs in exploring state, or $(i{-}1, j)$ and $(i, j{-}1)$ for MAVs in the returning state. Given pheromone levels of $\phi_{(i+1,j)}$ and $\phi_{(i,j+1)}$, the probability of a MAV choosing the left or right path is calculated using (1) to (4), where μ is a constant which determines the attractiveness of unexplored paths and is set to 0.75 for the simulations in [9]. If there is no MAV in the node state at (i, j) then $\phi_{(i,j)} = 0$. Equations (3) and (4) are the calculations of the probabilities of taking the left or right path at (i, j) where the correction factor $c_L(i, j)$ defined in [9] is applied to the original probability calculation $p_L(i, j)$ given in [6]. This correction ensures that positions equidistant from the base node have an equal chance of being eventually reached, given equal amounts of pheromone on every path.

$$p_L(i,j) = \frac{\left[\mu + \phi_{(i+1,j)}\right]^2}{\left[\mu + \phi_{(i+1,j)}\right]^2 + \left[\mu + \phi_{(i,j+1)}\right]^2} \tag{1}$$

$$c_L(i,j) = \frac{i+1}{i+j+2} \tag{2}$$

$$\pi_L(i,j) = \frac{p_L(i,j) \cdot c_L(i,j)}{p_L(i,j) \cdot c_L(i,j) + (1 - p_L(i,j)) \cdot (1 - c_L(i,j))} \tag{3}$$

$$\pi_R(i,j) = 1 - \pi_L(i,j) \tag{4}$$

3 Modelling the Scenario

Next we discuss the design and automatic generation of discrete, parameterised models of the scenario to which we can apply probabilistic analysis.

3.1 The PRISM Model Checker

Models of the scenario were constructed using the *probabilistic model checker* PRISM [11]. Given a probabilistic model of a system, PRISM can be used to analyse both temporal and probabilistic properties of the input model by exhaustively checking some logical requirement against all possible behaviours. Properties to be checked can be specified using *probabilistic temporal logics* such as Probabilistic Computation Tree Logic (PCTL) [8]. PCTL consists of classical logical operators $(\wedge, \vee, \neg)$, temporal operators $\Box\phi$ (at all points in the future ϕ holds), $\Diamond\phi$ (at some point in the future ϕ holds), $\phi\mathcal{U}\psi$ (ϕ holds until ψ holds), and the probabilistic operator $\mathrm{P}_{\bowtie\gamma}(\phi)$ where $\bowtie \in \{<, \leq, >, \geq\}$ is a relational operator and γ is a probability threshold. PCTL can therefore be used to specify properties such as $\mathrm{P}_{\geq 0.5}(\Diamond\phi)$, meaning "$\phi$ holds at some future point with a probability of at least 0.5". PRISM allows properties to be expressed which

evaluate to a numerical value, for instance $\mathrm{P}_{=?}(\Diamond\phi)$, "the probability of ϕ being true at some point in the future", and also supports metric temporal operators where the property is bound by time, for example $\Diamond^{\leq T}\phi$, "ϕ holds at some point in the future within T units of time".

3.2 Discretisation

Simulations were conducted in [9] using a time-step of $50\,ms$. Since a MAV travels at $10\,m/s$, and ideal positions for nodes are $100\,m$ distant from their neighbours, a MAV in the exploring or returning state takes $10\,s$ to move from one position to the next. In the probabilistic models we construct using PRISM we consider one transition in the model to be equivalent to a time-step of $10\,s$. In the original scenario MAVs are launched from the base node by a human operator every $15 \pm 7.5\,s$, giving an interval in seconds of possible durations between launches of $[7.5, 22.5]$. By rounding the endpoints of the interval to the nearest 10 (since one transition in our model is equivalent to $10\,s$), we determined that a MAV is launched from the base in our model once every 1 or 2 transitions. This non-determinism in the model resulted in state spaces too large to verify properties for, so the model was further simplified so that exactly one MAV was launched per transition. A comparison of the results obtained from applying probabilistic model checking to two models differing only by this additional simplification showed that there were only very minor differences between the results.

When a MAV switches to node state at time t and position (i, j) it initialises the pheromone level $\phi_{(i,j)}(t)$ to ϕ_{init}. The evolution of the pheromone levels at some position (i, j) is defined given by the equation

$$\phi_{(i,j)}(t+1) = \min[\phi_{i,j}(t) - \Delta\phi_{\text{dec}} + n \cdot \Delta\phi_{\text{ant}} + \Delta\phi_a, \phi_{\max}] \tag{5}$$

where $\phi_{\max}$ is the maximum amount of pheromone that can be deposited at any node, $\Delta\phi_{\text{dec}}$ is the rate at which deposited pheromone dissipates, $\Delta\phi_{\text{ant}}$ is the rate at which pheromone is deposited on a MAV in the node state by a MAV in the exploring state or the returning state, n is the number of MAVs in the exploring state or the returning state at (i, j), $\Delta\phi_a = \Delta\phi_{\text{int}}$ if there is some MAV in the node state at $(i{+}1, j)$ or $(i, j{+}1)$, or 0 otherwise, and $\Delta\phi_{\text{int}}$ is the rate at which extra pheromone is deposited on internal nodes.

The simulations conducted in [9] used values $\phi_{\text{init}} = 0.7$ and $\phi_{\max} = 1$. The rates at which pheromone was deposited, or dissipated, given in terms of units per time-step with one time-step corresponding to $50\,ms$, were given as $\Delta\phi_{\text{ant}} = 0.002$, $\Delta\phi_{\text{int}} = 0.001$ and $\Delta\phi_{\text{dec}} = 0.001$; in our model we consider one transition to be equivalent to $10\,s$ and therefore multiply these values by 200 to get the pheromone deposition/dissipation rates per transition in the model. To decrease the size of our models we use a range of discrete integer values to model pheromone levels. Given some $Ph \in \mathbb{N}$, the number of discrete values to be used to model pheromone levels, we define a mapping $\tau : [0, \phi_{\max}] \to \mathbb{N}$ that maps a pheromone value in $[0, \phi_{\max}]$ to some integer value such that $\tau(\phi) = [\phi \cdot Ph]$. In our models we use a value of $Ph = 5$; pheromone is deposited by MAVs in

the exploring state or the returning state at a rate of $\tau(200 \cdot \Delta\phi_{\text{ant}}) = 2$ per transition, internal nodes supplement their own pheromone levels at a rate of $\tau(200 \cdot \Delta\phi_{\text{int}}) = 1$, and pheromone dissipates at a rate of $\tau(200 \cdot \Delta\phi_{\text{dec}}) = 1$.

3.3 Abstractions and Assumptions

Modelling each MAV individually would result in intractable models. We therefore take advantage of the following abstractions. First, since one transition in the models is equivalent to the duration of a flight between two adjacent nodes, and since all MAVs begin at the base node (position $(0, 0)$, see Figure 1), after each transition we can assume that the location of each MAV is always at some position (i, j), instead of in-between positions. Second, as demonstrated in [12] a *counting abstraction* can also be used when modelling the behaviours of multiple identical processes. Since all MAVs are behaviourally identical, and their action decisions depend solely on their immediate state and percepts, when appropriate we can associate a counter with every position (i, j) that records the number of MAVs at that location.

In the simulations in [9] if the signal to land has not been given, then MAVs that have returned to the base node instead resume exploration. Here we constrain the number of MAVs that may leave the base node at any moment in time to a single MAV. This simplification allows us to greatly reduce the size of the model. Since each MAV is considered to land upon returning, and at most one MAV is launched each round, we can conclude that at most one exploring MAV is at any given position at any time. This can be modelled using Boolean variables to record if an exploring MAV has moved from some position (i, j) to either of $(i{+}1, j)$ or $(i, j{+}1)$.

A strategy is given in [9] to automatically assign altitudes to individual MAVs, ensuring that MAVs in the exploring or the returning state maintain an altitude higher than MAVs in the node state. While this strategy did not prove to be successful in all cases the chance of a collision occurring was sufficiently low (2.6% of 7500 MAVs collided over 500 trials) for us to assume that altitude differentiation always avoids collisions.

3.4 Prism Models

Each generated PRISM model can be defined as a set of m *modules* $M_1, \ldots, M_m$ [15]. Each module M_i is a tuple $(\mathcal{V}_i, \mathcal{I}_i, \mathcal{C}_i)$, where $\mathcal{V}_i = \{v_1, \ldots, v_{k_i}\}$ is a set of local variables over the domain consisting of finitely bound integers and booleans, $\mathcal{I}_i$ is a mapping of variables to initial values, and $\mathcal{C}_i = \{c_1, \ldots, c_{n_i}\}$ is a set of commands that define the behaviour of the module. With each local variable $v \in \mathcal{V}_i$ we associate a variable v' denoting the state of v in the next moment of time. The set of all local variables in the model is denoted as $\mathcal{V} = \bigcup_{i=1}^{m} \mathcal{V}_i$. For a module M_i every command $c_j \in \mathcal{C}_i$ is a pair $(g, \mathcal{U})$ where g is a predicate over $\mathcal{V}$ and $\mathcal{U} = \{(p_1, u_1), \ldots, (p_t, u_t)\}$ is a set of possible transitions for M_i. For a pair $(p_j, u_j) \in \mathcal{U}$, u_j is an assignment of values to each of the local variables

$v_1, \ldots, v_{k_i}$ and is of the form $\bigwedge_{a=1}^{k_i}(v'_a = ex_a)$ where ex_a is an expression in terms of V and the domain of variables, and $p_i \in \mathbb{R}^+$ is a constant defining the probability of that update occurring. Since our model is a DTMC it is required that for every command we have $p_i \in (0, 1]$ for $1 \leq i \leq t$, and $\sum_{i=1}^{t} p_i = 1$.

The semantics of a PRISM model can be defined in terms of a DTMC. A DTMC is a tuple $(\mathcal{S}, \bar{s}, \mathbf{P})$ where $\mathcal{S}$ is a set of states, $\bar{s} \in \mathcal{S}$ is the initial state, and $\mathbf{P} : \mathcal{S} \times \mathcal{S} \to [0, 1]$ is the probability transition matrix. The state space S_i of a module M_i is the set of all valuations of $\mathcal{V}_i$, and the global state space of a model is the product of the local state spaces of all modules. The local state of a module M_i is denoted as s_i, and a global state $s \in \mathcal{S}$ is a tuple $(s_1, \ldots, s_m)$ of local states. The initial state $\bar{s}$ is determined by $\mathcal{I}$. For brevity, we omit the details of the calculation of $\mathbf{P}$ and refer the reader to [15], noting that while many different parallel compositions of modules can be defined in PRISM, in our model all modules synchronise over all transitions.

3.5 Model Generation

We automate the generation of our models by using the PRISM Preprocessor to construct a parameterised model of the system. Then, given values for the parameters $N \in \mathbb{N}$, the number of MAVs in the swarm, $D \in \mathbb{N}$, the maximum distance between the target user and the base node (in hundreds of metres), and Ph, the number of discrete values used to record pheromone levels, we can automatically generate a PRISM model $\mathcal{M}$. The model is given by

$$\begin{aligned}\mathcal{M} = \{B\} &\cup \{E_{i,j}, R_{i,j} \mid i, j \in 0 \ldots D \text{ and } 0 < i + j < D\} \\ &\cup \{F_{i,j} \mid i, j \in 0 \ldots D \text{ and } i + j = D\},\end{aligned}$$

where B is a module that models the movement of MAVs in exploring state from the base node, each $E_{i,j}$ is a module that models the movement of MAVs in the exploring or node states at (i, j), each $R_{i,j}$ is a module that models the movement of MAVs in the returning state at (i, j), and each $F_{i,j}$ is a module that models the movement of MAVs in the exploring, node or returning states, at or beyond (i, j). We now define each module in the model, however due to space limitations we simply provide an informal description of the behaviour of each module.

The base module is a tuple $B = (\mathcal{V}_B, \mathcal{I}_B, \mathcal{C}_B)$, where $\mathcal{V}_B = \{bc, \nwarrow_{0,0}, \nearrow_{0,0}\}$ and $\mathcal{I}_B = \{bc \mapsto N, \nwarrow_{0,0} \mapsto \mathit{false}, \nearrow_{0,0} \mapsto \mathit{false}\}$. The finitely bound integer variable bc records the number of MAVs at the base node. The boolean variables $\nwarrow_{0,0}$ and $\nearrow_{0,0}$ record the movement of MAVs in the exploring state, and are *true* iff in the last moment in time a MAV moved from the base node to $(1, 0)$ and $(0, 1)$, respectively. Should there be one or more MAVs at the base node then one will be launched and will move to $(1, 0)$ with probability $\pi_L(0, 0)$, or to $(0, 1)$ with probability $\pi_R(0, 0)$.

For every $E_{i,j} = \{\mathcal{V}_E^{i,j}, \mathcal{I}_E^{i,j}, \mathcal{C}_E^{i,j}\}$ we have $\mathcal{V}_E^{i,j} = \{p_{i,j}, n_{i,j}, \nwarrow_{i,j}, \nearrow_{i,j}\}$ with $\mathcal{I}_E^{i,j} = \{p_{i,j} \mapsto 0, n_{i,j} \mapsto 0, \nwarrow_{i,j} \mapsto \mathit{false}, \nearrow_{i,j} \mapsto \mathit{false}\}$, where $p_{i,j}$ is a finitely bound integer variable recording the levels of pheromone deposited at (i, j), $n_{i,j}$ is a

boolean variable that is *true* if there is a MAV in the node state at (i, j), and $\nwarrow_{i,j}$ and $\nearrow_{i,j}$ are boolean variables which are are *true* iff a MAV moved from (i, j) respectively to $(i{+}1, j)$ or $(i, j{+}1)$ in the last moment in time. If no MAVs in the exploring state have moved to (i, j) then in the next moment in time no MAVs in the exploring state will be moving from (i, j); if $n_{i,j} = \mathit{true}$ then pheromone updates will be applied, and if $p_{i,j} \leq 0$ then in the next moment in time the MAV will be in the returning state, otherwise the MAV remains in the node state. If a MAV in the exploring state has moved to (i, j) when $n_{i,j} = \mathit{false}$ then in the next moment in time $n_{i,j}$ will be *true* and $p_{i,j}$ will be initialised to $\tau(\phi_{\text{init}})$; no MAV in the exploring state will be moving from (i, j) in the next moment in time. If a MAV in the exploring state has moved to (i, j) when $n_{i,j}$ = *true*, then in the next moment in time the exploring MAV will have moved to $(i{+}1, j)$ with probability $\pi_L(i, j)$, or to $(i, j{+}1)$ with probability $\pi_R(i, j)$, and remains in the exploring state; pheromone updates are applied and if $p_{i,j} \leq 0$ then in the next moment in time the MAV in the node state will be in the returning state, otherwise this MAV remains in the node state.

For every $R_{i,j} = \{\mathcal{V}_R^{i,j}, \mathcal{I}_R^{i,j}, \mathcal{C}_R^{i,j}\}$ if $i \geq 0, j > 0$ we have a finitely bound integer variable $\swarrow_{i,j} \in \mathcal{V}_R^{i,j}$ with $\mathcal{I}_R^{i,j}(\searrow_{i,j}) = 0$, and if $i > 0, j \geq 0$ we have a finitely bound integer variable $\searrow_{i,j} \in \mathcal{V}_R^{i,j}$ with $\mathcal{I}_R^{i,j}(\searrow_{i,j}) = 0$, which record the number of MAVs in the returning state that moved from (i, j) respectively to $(i, j{-}1)$ and $(i{-}1, j)$ in the last moment in time. Unlike exploring MAVs, it is often the case that two or more returning MAVs will simultaneously move to the same position. Any MAV in the returning state at some location (i, j) will always move respectively to $(i{-}1, j)$ or $(i, j{-}1)$ if $i = 0$ or $j = 0$, otherwise it will move from (i, j) to $(i{-}1, j)$ with probability $\pi_L(i{-}1, j{-}1)$, or to $(i, j{-}1)$ with probability $\pi_R(i{-}1, j{-}1)$. We define $E^{i,j}_{k,l}$ to be the event where given k MAVs in the returning state at (i, j), l MAVs move from (i, j) to $(i{-}1, j)$ in the next moment in time, and $k - l$ MAVs move from (i, j) to $(i, j{-}1)$ in the next moment in time. Given values for i, j, k and l we can calculate the probability of $E^{i,j}_{k,l}$ as

$$P(E^{i,j}_{k,l}) = \left(\pi_L(i-1, j-1)^l \cdot \pi_R(i-1, j-1)^{k-l}\right)\binom{k}{l} \ ,$$

since if we have k MAVs then we must consider all distinct subsets of size l. If no MAVs in the returning state have moved to (i, j), and there is no MAV in the node state whose pheromone levels have depleted, then in the next moment in time no MAVs in the returning state will be moving to either of $(i{-}1, j)$ or $(i, j{-}1)$. We then consider each case where there are k MAVs at (i, j) where $0 < k \leq N$. In each case calculating $P(E^{i,j}_{k,l})$ for $l = 0, \ldots, k$ gives a discrete probability distribution since for any $k \in \mathbb{N}$ we have that $\sum_{l=0}^{k} P(E^{i,j}_{k,l}) = 1$.

For every $F_{i,j} = \{\mathcal{V}_F^{i,j}, \mathcal{I}_F^{i,j}, \mathcal{C}_F^{i,j}\}$ we have a finitely bound integer variable $m_{i,j} \in \mathcal{V}_F^{i,j}$ with $\mathcal{I}_F^{i,j}(m_{i,j}) = 0$ that records the number of MAVs that are at or beyond this position and a finitely bound integer variable $r_{i,j} \in \mathcal{V}_F^{i,j}$ with $\mathcal{I}_F^{i,j}(r_{i,j}) = 0$ that is used to determine when MAVs should return from (i, j)

or beyond. If $i \geq 0, j > 0$ we have a finitely bound integer variable $\swarrow_{i,j} \in \mathcal{V}_F^{i,j}$ with $\mathcal{I}_F^{i,j}(\swarrow_{i,j}) = 0$ and if $i > 0, j \geq 0$ we have a finitely bound integer variable $\searrow_{i,j} \in \mathcal{V}_F^{i,j}$ with $\mathcal{I}_F^{i,j}(\searrow_{i,j}) = 0$ which record the number of MAVs in the returning state that moved from (i,j) respectively to $(i,j-1)$ and $(i-1,j)$ in the last moment in time. The variable $r_{i,j}$ is an approximation of the average number of transitions we would expect between a MAV in the exploring state moving to (i,j) or beyond and the first moment a MAV should return from a position at or beyond (i,j). If no MAV moves to (i,j) then after each transition the variable $r_{i,j}$ is decremented by 1. If there are MAVs at (i,j) and $r_{i,j}$ has decreased to 0, then in the next moment in time $m_{i,j}$ is decremented by 1 and a single MAV returns to $(i-1,j)$ or $(i,j-1)$ with probability respectively. For final positions where $i = 0$, all MAVs returning from (i,j) will move to $(i,j-1)$. Similarly for final positions where $j = 0$, all MAVs returning from (i,j) will move to $(i-1,j)$.

4 Experiments

To validate our model we applied statistical model checking using the PRISM discrete-event simulator and compared our results to those obtained from the simulations conducted in [9]. The mean probability of establishing contact with a user within 30 minutes was calculated over a series of 500 simulations for varying swarm sizes. Users were located at some randomly determined location within a 60 degree arc in a known cardinal direction from the base node at a distance of ≈200-500 m. In our model we assumed that a MAV has established communication with a user if it has moved to a position at most 100 m distant from the user. Since a user can be located up to ≈500 m from the base node, models were generated with $D = 5$ and $Ph = 5$ for all models. We define $\mathcal{P}_{user} = \{(i,j) \mid i,j \in 0 \ldots 5 \text{ and } 1 < i{+}j \leq 5\}$ to be the set of all possible locations at which a user may be located. For each $(i,j) \in \mathcal{P}_{user}$ we used PRISM to calculate the probability of a MAV moving to (i,j) within 30 minutes (equivalent to 180 transitions in our model) by formally specifying this as a probabilistic reachability property in PCTL. Since the grid of positions is symmetrical we only check PCTL properties for a subset of $\mathcal{P}_{user}$, as shown in Figure 3; the probability of a MAV moving to some (i,j) is equivalent to the probability of a MAV moving to (j,i). For each (i,j) we define $moved_{i,j}$ as

$$moved_{i,j} \equiv \begin{cases} (\nwarrow_{i-1,j} \vee \nearrow_{i,j-1}) & \text{if } i,j > 0 \\ \nearrow_{i,j-1} & \text{if } i = 0, j > 0 \\ \nwarrow_{i-1,j} & \text{if } i > 0, j = 0 \end{cases} \tag{6}$$

We then calculate λ, the mean probability over all locations, as

$$\lambda = \left(\sum_{(i,j) \in \mathcal{P}_{user}} \mathrm{P}_{=?}(\Diamond^{\leq 180} moved_{i,j}) \right) / \, |\mathcal{P}_{user}| \tag{7}$$

Figure 4 compares the results of calculating λ for values of $N \in 5, \ldots, 20$ to the results presented in [9]. Statistical model checking results were obtained using 500 discrete-event simulation samples with an average confidence interval of ±2%

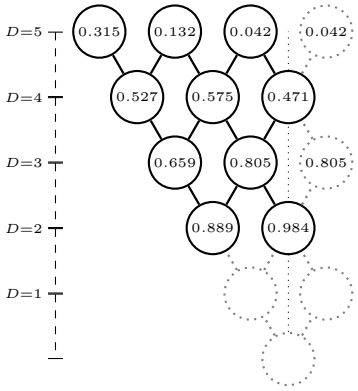

Fig. 3. The subset of $\mathcal{P}_{user}$ for which the PCTL property is checked. Here the results correspond to checking the property for $N = 5$.

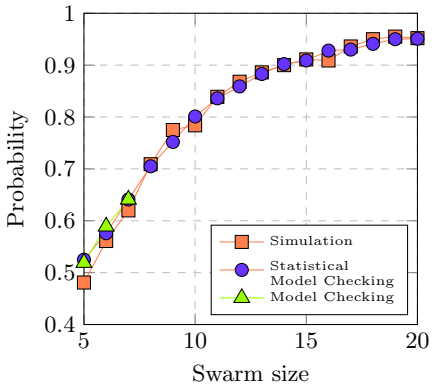

Fig. 4. The mean probability of finding the user within 30 minutes over 500 trials for simulation and 500 samples for statistical model checking.

based on a 99.0% confidence level. Experiments were conducted on a PC with a 2.20GHz Intel Xeon E5-2420 CPU, 196GB RAM, running Scientific Linux 6.6. There is clearly a strong correlation between both sets of results. For some swarm sizes, namely for $N = 5, 9, 10, 16$, there was a more pronounced difference between the two values. However, some minor discrepancies were expected due to the relatively low number of simulations/samples used to obtain the results. Exhaustive model checking results are shown only for $N = 5, 6, 7$ since the reachable state space of models for $N \geq 8$ was too large to be calculated.

PRISM can be used to reason about other measurable aspects of model behaviours. Rewards can be associated with individual states, or groups of states, and properties relating to expected values for these rewards can be checked in models. PRISM also provides the R operator which allows properties to be expressed such as the reachability reward property $\mathrm{R}_{=?}(\Diamond\phi)$, "what is the expected reward for reaching a state where ϕ is true". By associating a reward of one with each state in our models we can test the property $\mathrm{R}_{=?}(\Diamond m_{i,j})$ for every $(i, j) \in \mathcal{P}_{user}$, which calculates the total time expected for the swarm to establish contact with a user at (i, j) with probability 1. These calculated values could facilitate the logistics of MAV swarm deployments where guaranteed contact with a user is required, given a limited number of MAVs. In Figure 5 the four graphs on the left show the total expected time in hours for a deployment of N MAVs, depth D, and lateral distance of the target user from the base node, to establish communication with the target user with a probability 1. This is done by checking the property $\mathrm{R}_{=?}(\Diamond moved_{i,j})$ for each (i, j) The four graphs on the right show the probability of establishing communication with the target user within 30 minutes by a deployment of N MAVs, depth D, and lateral distance of the target user from the base node. This is done by checking the property

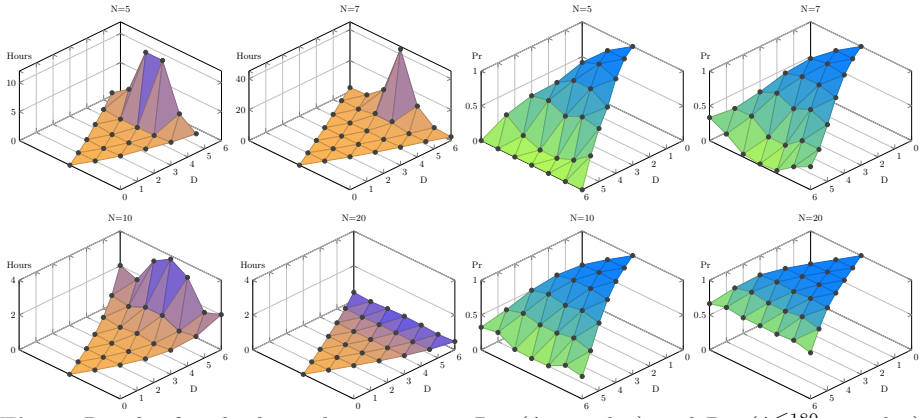

Fig. 5. Results for checking the properties $\mathrm{R}_{=?}(\Diamond moved_{i,j})$ and $\mathrm{P}_{=?}(\Diamond^{\leq 180} moved_{i,j})$ for each (i, j) in the generated models.

$\mathrm{P}_{=?}(\Diamond^{\leq 180} moved_{i,j})$ for each (i, j). Results where $N > 7$ were obtained using statistical methods over 4000 samples.

5 Conclusions and Further Work

We have constructed formal probabilistic models making some simplifying assumptions given in Section 3.3, and clearly shown a close correspondence between these models and the simulations conducted in the original scenario. We then used these models to verify both probabilistic and reward-based properties, where the resultant calculated values could be used to plan the deployment of a swarm of MAVs where establishing contact with a user must be guaranteed, or achieved with a probability that exceeds some given threshold. Since battery life greatly impacts the flight duration of MAVs the a priori calculation of the total expected flight time, or total expected distance travelled, for the swarm would ensure that sufficient resources could be made available to ensure that it achieves its objectives.

A natural extension of this work would be to use the parametric model checking functionality of Prism to investigate more thoroughly how each parameter affects the results of checking properties in the model and to calculate optimum parameter values that maximise or minimise probabilistic or reward-based properties. We also aim to further abstract our approach so that the techniques that we have developed here can be applied to a broader range of swarm algorithms where stigmergic communication is used to coordinate the behaviour of the swarm.

6 Acknowledgments

The authors would like to thank the Networks Sciences and Technology Initiative (NeST) of the University of Liverpool for the use of their computing facilities.

The first author would like to acknowledge the funding received from the Sir Joseph Rotblat Alumni Scholarship.

References

1. Behdenna, A., Dixon, C., Fisher, M.: Deductive verification of simple foraging robotic behaviours. International Journal of Intelligent Computing and Cybernetics 2(4), 604–643 (2009)
2. Beni, G.: From swarm intelligence to swarm robotics. In: SAB 2004 Revised Selected Papers, LNCS, vol. 3342, pp. 1–9. Springer (2005)
3. Bonabeau, E., Dorigo, M., Theraulaz, G.: Swarm intelligence: from natural to artificial systems. Oxford University Press (1999)
4. Bonabeau, E., Sobkowski, A., Theraulaz, G., Deneubourg, J.L.: Adaptive task allocation inspired by a model of division of labor in social insects. In: Proc. BCEC97. pp. 36–45. World Scientific (1997)
5. Campo, A., Dorigo, M.: Efficient multi-foraging in swarm robotics. In: Proc. ECAL 2007. LNCS, vol. 4648, pp. 696–705. Springer (2007)
6. Deneubourg, J.L., Goss, S., Franks, N., Pasteels, J.M.: The blind leading the blind: modeling chemically mediated army ant raid patterns. Journal of Insect Behavior 2(5), 719–725 (1989)
7. Dixon, C., Winfield, A.F., Fisher, M., Zeng, C.: Towards temporal verification of swarm robotic systems. Robotics and Autonomous Systems 60(11), 1429–1441 (2012)
8. Hansson, H., Jonsson, B.: A logic for reasoning about time and reliability. Formal Aspects of Computing 6(5), 512–535 (1994)
9. Hauert, S., Winkler, L., Zufferey, J.C., Floreano, D.: Ant-based swarming with positionless micro air vehicles for communication relay. Swarm Intelligence 2(2-4), 167–188 (2008)
10. Herd, B., Miles, S., McBurney, P., Luck, M.: Approximate verification of swarm-based systems: a vision and preliminary results. In: Proc. 23rd Safety-critical Systems Symposium. pp. 361–378. SCSC, SCSC (2015)
11. Hinton, A., Kwiatkowska, M., Norman, G., Parker, D.: Prism: A tool for automatic verification of probabilistic systems. In: Proc. TACAS 2006, LNCS, vol. 3920, pp. 441–444. Springer (2006)
12. Konur, S., Dixon, C., Fisher, M.: Analysing robot swarm behaviour via probabilistic model checking. Robotics and Autonomous Systems 60(2), 199–213 (2012)
13. Kouvaros, P., Lomuscio, A.: Verifying emergent properties of swarms. In: Proc. AAAI 2015. pp. 1083–1089. AAAI Press (2015)
14. Liu, W., Winfield, A., Sa, J., Chen, J., Dou, L.: Strategies for energy optimisation in a swarm of foraging robots. In: SAB 2007 Revised Selected Papers. LNCS, vol. 4433, pp. 14–26. Springer (2007)
15. Parker, D.A.: Implementation of symbolic model checking for probabilistic systems. Ph.D. thesis, University of Birmingham (2002)
16. Şahin, E., Winfield, A.: Special issue on swarm robotics. Swarm Intelligence 2(2), 69–72 (2008)
17. Støy, K.: Using situated communication in distributed autonomous mobile robotics. In: Proc. SCAI 2001. pp. 44–52. IOS Press (2001)
18. Theraulaz, G., Bonabeau, E.: Coordination in distributed building. Science 269, 686–686 (1995)